sklearn.metrics.precision_recall_fscore_support

» sklearn.metrics.**precision_recall_fscore_support**(*y_true*, *y_pred*, *beta=1.0*, *labels=None*, *pos_label=1*, *average=None*, *warn_for=('precision', 'recall', 'f-score')*, *sample_weight=None*) [source]

Compute precision, recall, F-measure and support for each class

The precision is the ratio `tp / (tp + fp)` where `tp` is the number of true positives and `fp` the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

The recall is the ratio `tp / (tp + fn)` where `tp` is the number of true positives and `fn` the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

The F-beta score can be interpreted as a weighted harmonic mean of the precision and recall, where an F-beta score reaches its best value at 1 and worst score at 0.

The F-beta score weights recall more than precision by a factor of `beta`. `beta == 1.0` means recall and precision are equally important.

The support is the number of occurrences of each class in `y_true`.

If `pos_label is None` and in binary classification, this function returns the average precision, recall and F-measure if `average` is one of `'micro'`, `'macro'`, `'weighted'` or `'samples'`.

Read more in the User Guide.

Parameters: **y_true** : 1d array-like, or label indicator array / sparse matrix

Ground truth (correct) target values.

y_pred : 1d array-like, or label indicator array / sparse matrix

Estimated targets as returned by a classifier.

beta : float, 1.0 by default

The strength of recall versus precision in the F-score.

labels : list, optional

The set of labels to include when `average != 'binary'`, and their order if `average is None`. Labels present in the data can be excluded, for example to calculate a multiclass average ignoring a majority negative class, while labels not present in the data will result in 0 components in a macro average. For multilabel targets, labels are column indices. By default, all labels in `y_true` and `y_pred` are used in sorted order.

pos_label : str or int, 1 by default

The class to report if `average='binary'`. Until version 0.18 it is necessary to set `pos_label=None` if seeking to use another averaging method over binary targets.

»

average : string, [None (default), 'binary', 'micro', 'macro', 'samples', 'weighted']

If `None`, the scores for each class are returned. Otherwise, this determines the type of averaging performed on the data:

`'binary'`:
: Only report results for the class specified by `pos_label`. This is applicable only if targets (`y_{true,pred}`) are binary.

`'micro'`:
: Calculate metrics globally by counting the total true positives, false negatives and false positives.

`'macro'`:
: Calculate metrics for each label, and find their unweighted mean. This does not take label imbalance into account.

`'weighted'`:
: Calculate metrics for each label, and find their average, weighted by support (the number of true instances for each label). This alters 'macro' to account for label imbalance; it can result in an F-score that is not between precision and recall.

`'samples'`:
: Calculate metrics for each instance, and find their average (only meaningful for multilabel classification where this differs from `accuracy_score`).

Note that if `pos_label` is given in binary classification with *average != 'binary'*, only that positive class is reported. This behavior is deprecated and will change in version 0.18.

warn_for : tuple or set, for internal use

This determines which warnings will be made in the case that this function is being used to return only one of its metrics.

sample_weight : array-like of shape = [n_samples], optional

Sample weights.

Returns:	**precision: float (if average is not None) or array of float, shape = [n_unique_labels]** :
	recall: float (if average is not None) or array of float, , shape = [n_unique_labels] :
	fbeta_score: float (if average is not None) or array of float, shape = [n_unique_labels] :
	support: int (if average is not None) or array of int, shape = [n_unique_labels] :
	The number of occurrences of each label in `y_true`.

»

References

[R175] Wikipedia entry for the Precision and recall

[R176] Wikipedia entry for the F1-score

[R177] *Discriminative Methods for Multi-labeled Classification Advances in Knowledge Discovery and Data Mining (2004), pp. 22-30 by Shantanu Godbole, Sunita Sarawagi <http://www.godbole.net/shantanu/pubs/multilabelsvm-pakdd04.pdf>*

Examples

```
>>> from sklearn.metrics import precision_recall_fscore_support
>>> y_true = np.array(['cat', 'dog', 'pig', 'cat', 'dog', 'pig'])
>>> y_pred = np.array(['cat', 'pig', 'dog', 'cat', 'cat', 'dog'])
>>> precision_recall_fscore_support(y_true, y_pred, average='macro')
...
(0.22..., 0.33..., 0.26..., None)
>>> precision_recall_fscore_support(y_true, y_pred, average='micro')
...
(0.33..., 0.33..., 0.33..., None)
>>> precision_recall_fscore_support(y_true, y_pred, average='weighted')
...
(0.22..., 0.33..., 0.26..., None)
```

It is possible to compute per-label precisions, recalls, F1-scores and supports instead of averaging: >>> precision_recall_fscore_support(y_true, y_pred, average=None, ... labels=['pig', 'dog', 'cat']) ... # doctest: +ELLIPSIS,+NORMALIZE_WHITESPACE (array([0. , 0. , 0.66...]),

array([0., 0., 1.]), array([0. , 0. , 0.8]), array([2, 2, 2]))